Verteilte Systeme 1

Technologien des World Wide Web

christian.zirpins@hs-karlsruhe.de

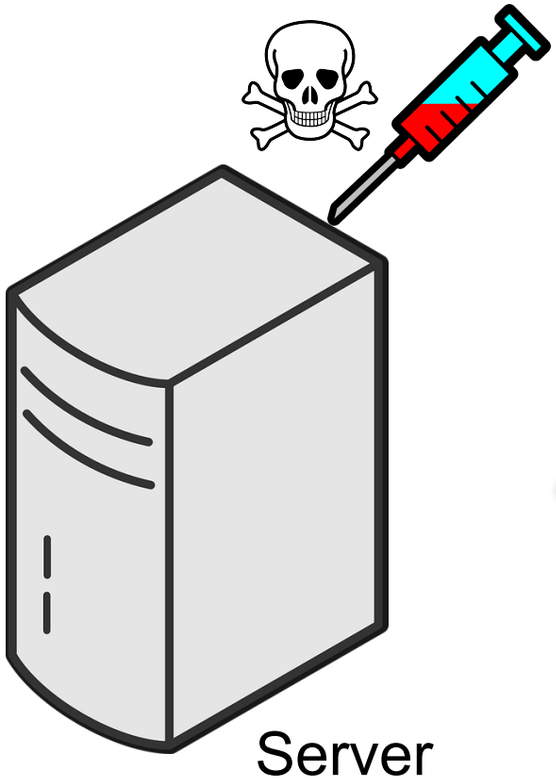
Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

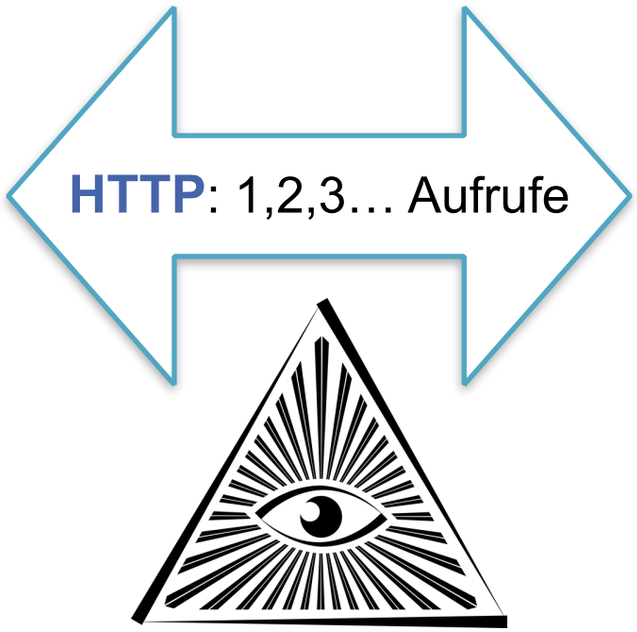
Web App Sicherheit - eine kurze Einführung

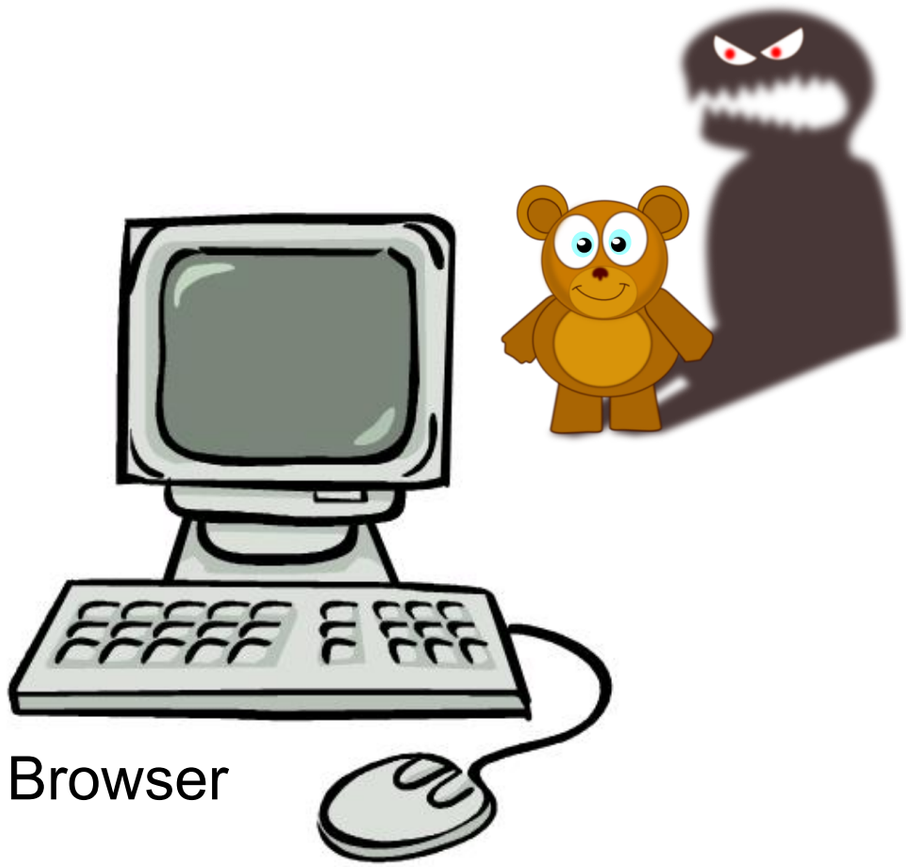
Heutige Lernziele

Nach dieser Vorlesung können Sie…

- …die häufigsten **Sicherheitsprobleme** von Web Anwendungen benennen.
- …einfache **Attacken** beschreiben, die gegen ungesicherten Code durchgeführt werden können.
- …**Maßnahmen** implementieren, die vor solchen Attacken schützen.







Web Apps sind ein beliebtes Angriffsziel…

Große Angriffsfläche

- Ein Angreifer kann an verschiedenen Punkten ansetzen
  - Web Server
  - Web Browser
  - Web Anwendung
  - Web User
- Web Anwendungen können Millionen Benutzer haben (es lohnt sich, diese zu "Hacken")
- Es gibt automatische Methoden (und konkrete Werkzeuge) um bekannte Schwachstellen in Web Servern/Apps zu suchen/testen

Web Anwendungen sind einfach zu entwickeln aber schwierig zu sichern.

Bedrohungen

- **Entstellung** (Defacement)
  - Das Aussehen einer Webseite ändern/ersetzen
- **Offenlegung von Daten**
  - Benutzerdatenbanken, Kreditkartennummern etc.
- **Datenverlust**
  - Angreifer löschen Daten
- **Unautorisierter Zugriff**
  - Angreifer können Funktionen einer Web App nutzen, die nicht für sie nutzbar sein sollten
- **Dienstleistungsverhinderung** (Denial of Service - DOS)
  - Eine Web App ist nicht mehr für legitime User nutzbar
- **“Fuß in der Tür”**
  - Angreifer gelangen in das interne Netz

Vortrag über Web Sicherheit vom CERN

Source: http://cds.cern.ch/record/1562545

Beispiel… healthcare.gov

WASHINGTON (AP) — Hackers successfully breached HealthCare.gov, but no consumer information was taken from the health insurance website that serves more than 5 million Americans, the Obama administration disclosed Thursday.

Instead, the hackers **installed malicious software** that could have been used to launch an attack on other websites from the federal insurance portal.

Health and Human Services spokesman Aaron Albright said **the website component that was breached had been used for testing and did not contain consumer information**, such as names, birth dates, Social Security numbers and income details.

Testen in der Wildnis…

The initial intrusion took place July 8, but it was not detected until Monday of last week during a **manual scan of system logs**. HHS said **the component that was breached did not have a firewall, or intrusion detection software, installed on it**. Technicians manually scanning logs discovered the breach Aug. 25 and took action.

ein einfaches Ziel

Source: https://www.yahoo.com/news/hackers-break-healthcare-gov-233616181.html?ref=gs

Beispiel… cern.ch

Statt das Ziel direkt anzugreifen, suche eine nahegelegene Schwachstelle

Hackers broke into a computer system at CERN's Large Hadron Collider, **targeting a system that was "one step away" from a control computer**, but otherwise appear to have done no major damage, according to a report on Friday in the British newspaper The Telegraph.

The system that was breached monitors the Compact Muon Solenoid Experiment, which will be analyzing data during subatomic particle collisions in the particle accelerator located along the French-Swiss border. Experiments, which began on Wednesday, are designed to help scientists explore particle physics theories.

During the attack on Tuesday and Wednesday, **hackers left behind half a dozen files, damaged one CERN file, and displayed a Web page with the headline "GST: Greek Security Team,"** signing off: "We are 2600--don't mess with us," (sic) CERN scientists told the newspaper.

Source: http://www.cnet.com/news/hackers-break-into-large-hadron-collider-computer/

Beispiel… www.nrc-cnrc.gc.ca

Hackers used tempting emails, malware and password theft to worm their way into National Research Council computers in pursuit of valuable scientific and trade secrets, a newly released federal analysis reveals.

…

Einfach zu erlangende Informationen

The cyber response centre's report details the "exploitation cycle" of the attack, saying it began with the **collection of valid email addresses for** research council employees. Messages containing malicious links **were then sent to the employees' inboxes** — a tactic known as spear phishing.

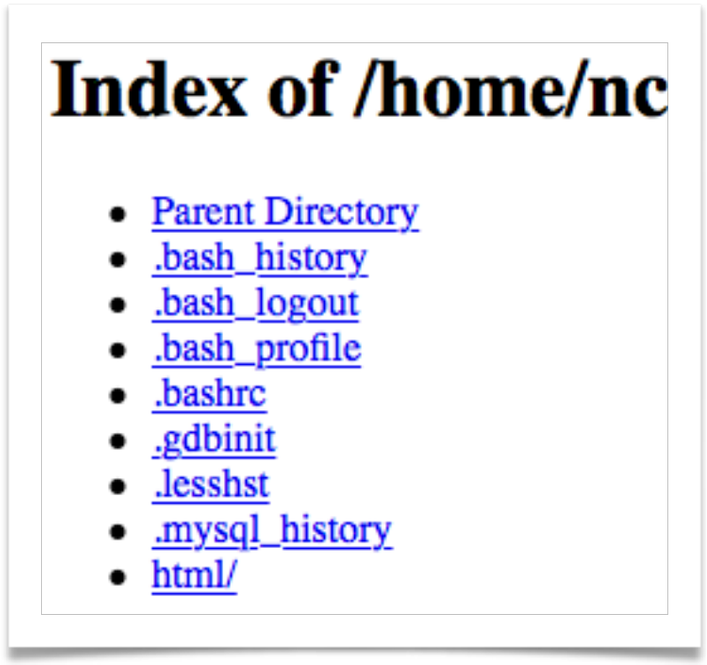
Es reicht, wenn ein Mitarbeiter den Link klickt…

Those who unwittingly clicked on the innocent-looking links set the next phase in motion by leading them to what cyber-sleuths call a "**watering hole website**" — a site of likely interest to people working in a specific organization or industry.

Quelle: http://www.cbc.ca/news/technology/chinese-hackers-installed-malware-on-national-research-council-computers-1.2872385

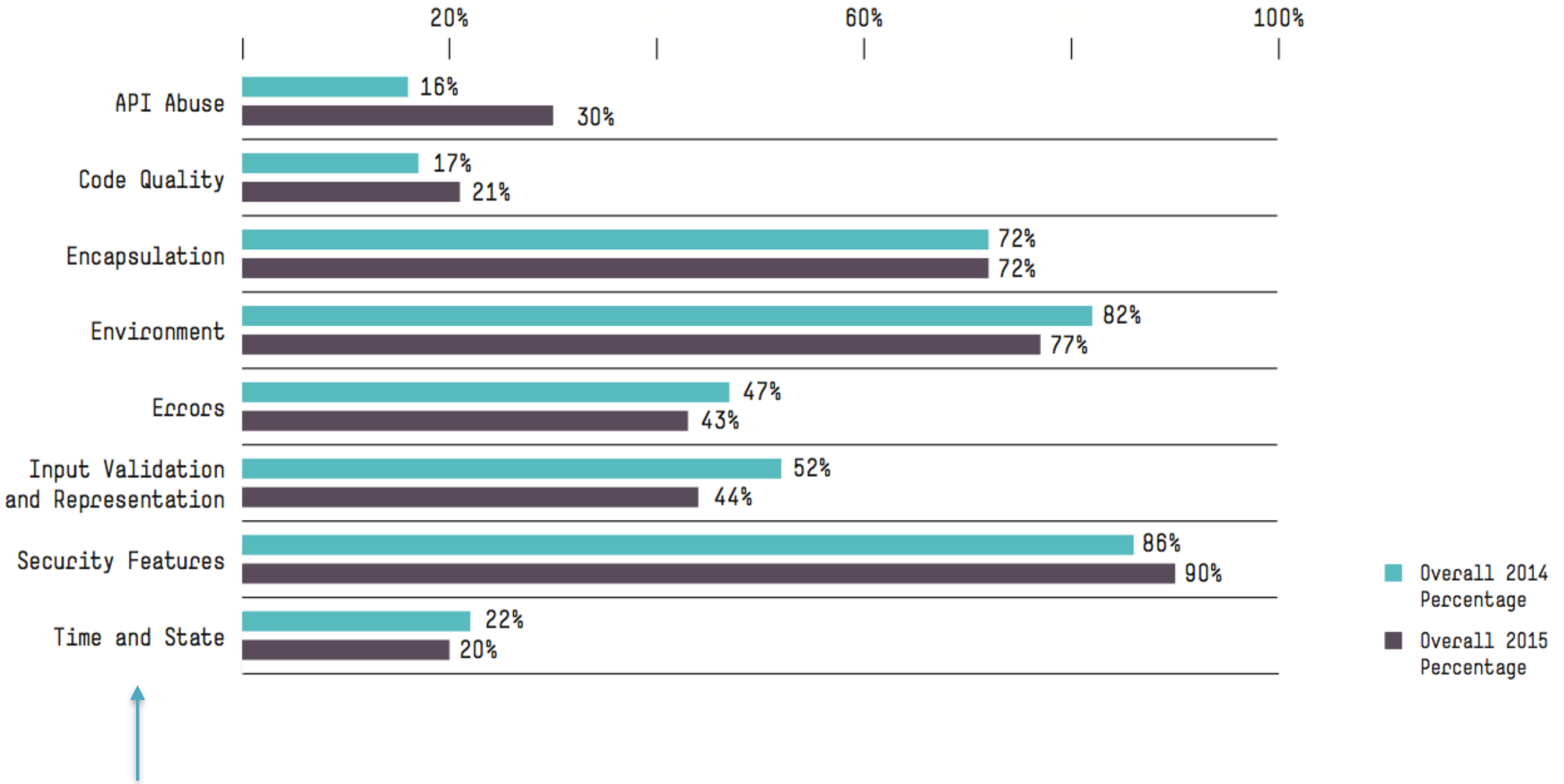
Sicherheitslücken im Web finden ist einfach

- Suchmaschinen stellen praktische Operatoren bereit, um Dateien zu finden, die wertvolle Informationen enthalten können (und durch Fehler öffentlich einsehbar sind)
  - `intitle:"index of" .bash_history`
  - `-inurl:https login`
  - Serverseitige Fehlermeldungen
- Bekannt als "Google Hacking"



Anwendungssicherheit

Wahrscheinlichkeit von Schwächen in Web Apps



"Pernicious Kingdoms"

HPE Security Research Cyber Risk Report 2016

Ein konkretes Beispiel für Bedrohung Nr. 1

In Kürze

- Web Apps, die **Benutzereingaben** erlauben, sind verwundbar
- Böse Nutzer 😈 können **HTML Code** (anstatt Text) in ein Formular eingeben (oder in ein editierbares HTML Element)
- Der hinzugefügte Code kann die **Erscheinung der Web App stark verändern**
  - **Andere Nutzer** geben vielleicht sensitive Informationen preis
  - **Angreifer** können diese Informationen ausspähen

Beispiel

```
var express = require("express");
var url = require("url");
var http = require("http");
var app;

var port = process.argv[2];
app = express();
http.createServer(app).listen(port);

app.get("/hello", function (req, res) {
  var query = url.parse(req.url, true).query;
  var name = (query["name"]!=undefined) ? query["name"] :
    "Anonymous";
   res.send("<html><head></head><body><h1>Greetings " + name +
    ", have a nice day!</h1></body></html>");
});
```

Web Server **prüft** die Benutzereingabe nicht

Beispiel

- Nicht alle Benutzer werden nur ihren Namen eingeben...
- Wie wäre es hiermit?

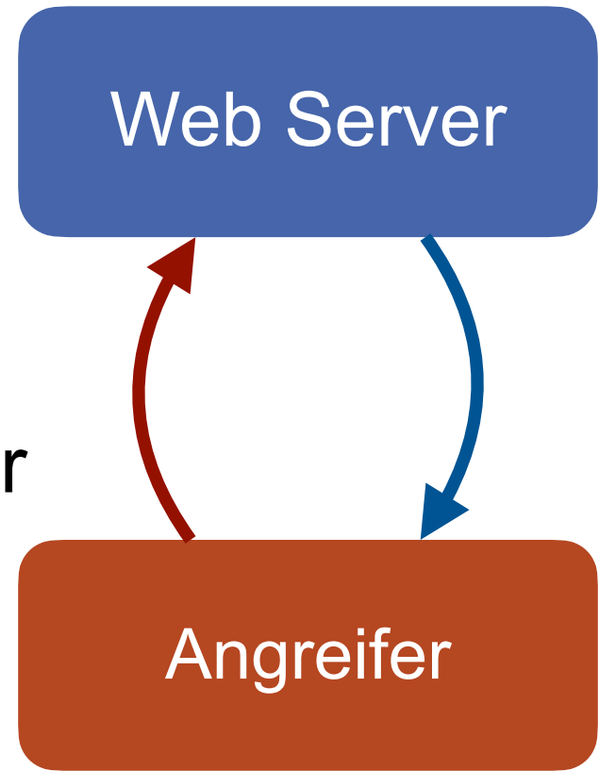
```
<h3>Please enter your name and password:</h3>
<form method="GET"
  action="http://127.0.0.1:4444/login">
Username:
<input type="text" name="username"/><br/>
Password:
<input type="password" name="password"/><br/>
<input type="submit" value="Login" />
</form>
<!--
```

Ein Server des Angreifers

Wozu dient das?

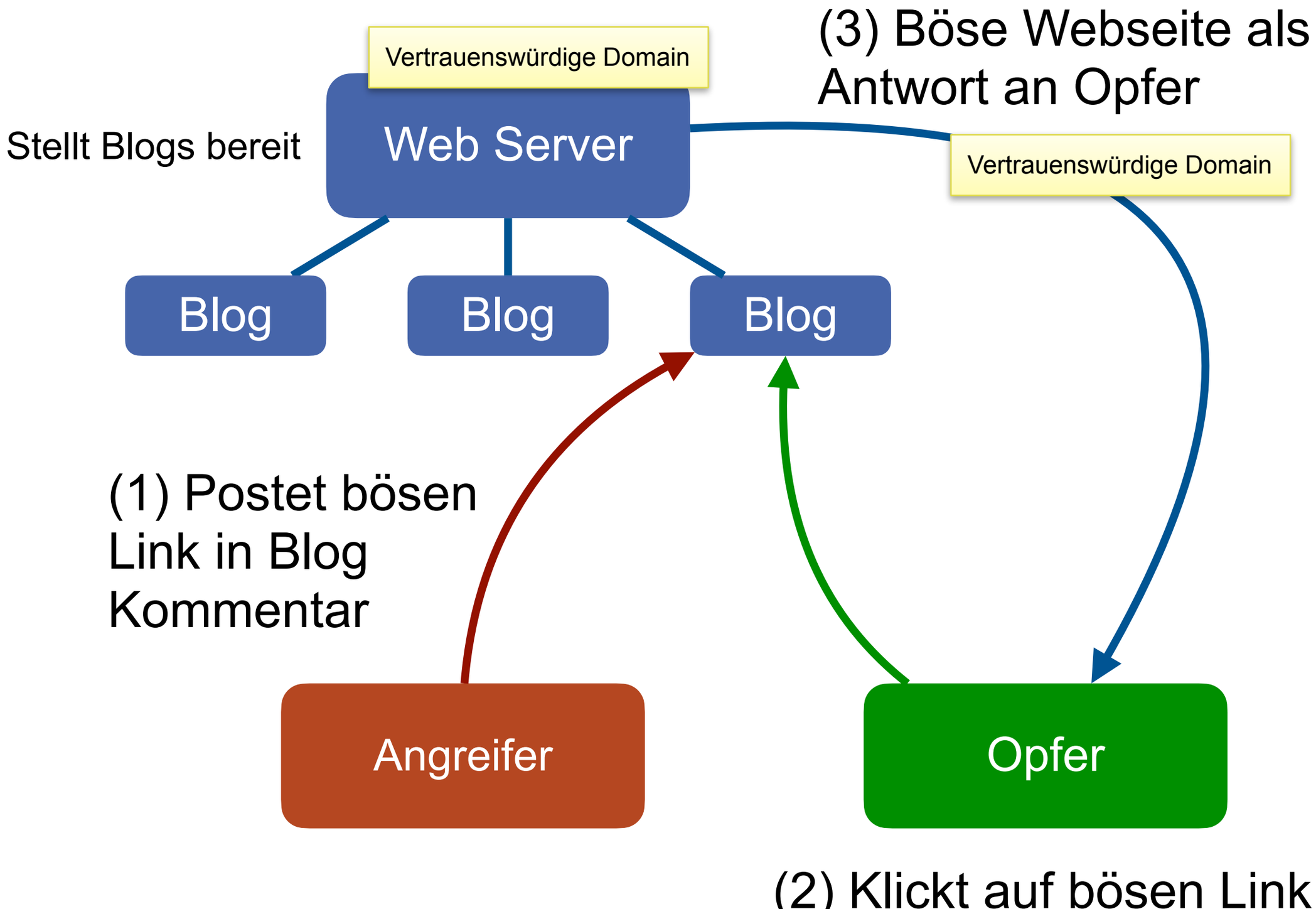
Moment mal, was bringt das schon?

GET
mit bösem
URL Parameter



Böse Webseite als
Antwort an Angreifer ?!

Eine Möglichkeit…



Was man gegen das Problem tun kann

- Verwende serverseitige Skripte um **alle** Benutzereingaben zu "**desinfizieren**" und die Ausgaben zu **kodieren**.
- Optionen:
  - Verwerfe alle HTML-Tags in der Benutzereingabe mithilfe eines regulären Ausdrucks
  - Lehne alle Eingaben ab, die "<" oder ">" enthalten
  - Kodiere HTML Einträge ("Escaping")

Es gibt für diese Aufgabe verschiedene node.js Module

```
var validator = require('validator');
...
var name = ( query["name"]!=undefined) ? query["name"] : "";
var cleaned = validator.escape(name); //escaping HTML
```

HTML Escaping

```
<h3>Please enter your name and password:</h3>
<form method="GET"
  action="http://127.0.0.1:4444/login">

Username:
<input type="text" name="username"/><br/>
Password:
<input type="password" name="password"/><br/>
<input type="submit" value="Login" />
</form>
<!–
```

HTML String "Unescaped"

```
&lt;h3&gt;Please enter your name and
password:&lt;/h3&gt;
&lt;form method=&quot;GET&quot;
  action=&quot;http://127.0.0.1:4444/
login&quot;&gt;

Username:
&lt;input type=&quot;text&quot;
name=&quot;username&quot;/&gt;&lt;br/&gt;
Password:
&lt;input type=&quot;password&quot;
name=&quot;password&quot;/&gt;&lt;br/&gt;
&lt;input type=&quot;submit&quot;
value=&quot;Login&quot; /&gt;
&lt;/form&gt;
&lt;!--
```

HTML String "Escaped"

- Escaping ersetzt alle Zeichen, die als Markup interpretiert werden könnten.
- Reservierte Zeichen werden durch **HTML Entities** ersetzt
  - " wird zu `&quot;`
  - & wird zu `&`
  - < wird zu `<`
  - > wird zu `>`

Etwas genereller…

Ungeprüfte Eingaben ausnutzen

1. Böse Daten in Web App injizieren
2. Manipuliere mit den bösen Daten die Anwendung

- Parameter-Manipulation in HTML Formularen
- URL-Manipulation (URLs beinhalten oft Parameter)
- Manipulation versteckter HTML-Felder
- HTTP-Header Manipulation
- Cookie Manipulation

Anwendungen manipulieren

- **SQL-Injektion**
  - Leite Eingabe mit SQL Kommandos an einen Datenbankserver zur Ausführung weiter
- **Cross-Site Scripting**
  - Nutze Anwendungen aus, die ungeprüfte Eingaben "wortwörtlich" ausgeben, um Benutzer dazu zu bringen, bösen Code auszuführen
- **Pfadumgehung**
  - Nutze ungeprüfte Benutzereingaben aus, um Zugriff von Dateien auf dem Server zu steuern
- **Kommando-Injektion**
  - Nutze ungeprüfte Benutzereingaben aus, um Shell Kommandos auszuführen.

Zur Vertiefung: OWASP TOP 10

- **Open Web Application Security Project** (OWASP)
  - https://owasp.org/www-project-top-ten/
- **TOP 10**: Liste der 10 kritischsten Sicherheitsrisiken für Web Apps[1]
  1. Injektion
  2. Fehlerhafte Authentifizierung und Session Management
  3. Preisgabe sensitiver Daten
  4. XML External Entities (XXE)
  5. Fehlerhafte Zugriffssteuerung für Funktionen und Daten
  6. Fehlerhafte Sicherheitskonfiguration
  7. Cross-site scripting (XSS)
  8. Unsichere Deserialisierung
  9. Komponenten mit bekannten Schwächen verwenden
  10. Ungenügendes Logging & Monitoring

[1]Version Sommer 2020 (Daten aus 2017)

Zusammenfassung

Zusammenfassung

- Web Anwendungen eröffnen viele Angriffsmöglichkeiten
- Eine Web Anwendung zu sichern setzt viel Wissen in unterschiedlichen Gebieten vorraus
- Wichtigste Message: **testen, testen, testen**
- Konzentrieren Sie sich zur Absicherung ihrer Anwendung auf die wichtigsten Angriffstypen (OWASP Top 10)

Literatur

Zur Vertiefung

- Hewlett Packard Enterprise, HPE Security Research - Cyber Risk Report 2016, http://www8.hp.com/de/de/software-solutions/cyber-risk-report-security-vulnerability/index.html
- Katrina Tsipenyuk, Brian Chess, Gary McGraw, "Seven Pernicious Kingdoms: A Taxonomy of Software Security Errors", https://cwe.mitre.org/documents/sources/SevenPerniciousKingdoms.pdf
- Open Web Application Security Project (OWASP), https://owasp.org/